```
/*
Display.cpp - A simple track GPS to SD card logger. Display module.
TinyTrackGPS v0.12

Copyright © 2019-2021 Francisco Rafael Reyes Carmona.
All rights reserved.

rafael.reyes.carmona@gmail.com

  This file is part of TinyTrackGPS.

  TinyTrackGPS is free software: you can redistribute it and/or modify
  it under the terms of the GNU General Public License as published by
  the Free Software Foundation, either version 3 of the License, or
  (at your option) any later version.

  TinyTrackGPS is distributed in the hope that it will be useful,
  but WITHOUT ANY WARRANTY; without even the implied warranty of
  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
  GNU General Public License for more details.

  You should have received a copy of the GNU General Public License
  along with TinyTrackGPS.  If not, see <https://www.gnu.org/licenses/>.
*/

#include "Display.h"

Display::Display(Display_Type t):_screen(t){
    _width = 16;
    _height = (_screen > 0) ? 2 : 8;
}

void Display::start(){
    #if defined(DISPLAY_TYPE_LCD_16X2)
        lcd = new LiquidCrystal(LCD_RS, LCD_ENABLE, LCD_D0, LCD_D1, LCD_D2, LCD_D3);
        lcd->begin(_width, _height);
    #elif defined(DISPLAY_TYPE_LCD_16X2_I2C)
        lcd = new LiquidCrystal_I2C(I2C,_width,_height);
        lcd->init();
        lcd->backlight();
    #endif

    #if defined(DISPLAY_TYPE_LCD_16X2) || defined(DISPLAY_TYPE_LCD_16X2_I2C)
        // DEFINICION DE CARACTERES PERSONALIZADOS
        static byte alt[8] = { 0x04, 0x0E, 0x1F, 0x04, 0x04, 0x04, 0x04, 0x04 };
        static byte ant[8] = { 0x0E, 0x11, 0x15, 0x11, 0x04, 0x04, 0x0E, 0x00 };
        static byte sd[8] = { 0x0E, 0x11, 0x1F, 0x00, 0x00, 0x17, 0x15, 0x1D };
        static byte hourglass_0[8] = { 0x1F, 0x0E, 0x0E, 0x04, 0x04, 0x0A, 0x0A,
0x1F };
        static byte hourglass_1[8] = { 0x1F, 0x0A, 0x0E, 0x04, 0x04, 0x0A, 0x0A,
0x1F };
        static byte hourglass_2[8] = { 0x1F, 0x0A, 0x0E, 0x04, 0x04, 0x0A, 0x0E,
0x1F };
        static byte hourglass_3[8] = { 0x1F, 0x0A, 0x0A, 0x04, 0x04, 0x0A, 0x0E,
0x1F };
        static byte hourglass_4[8] = { 0x1F, 0x0A, 0x0A, 0x04, 0x04, 0x0E, 0x0E,
0x1F };
        lcd->createChar(0, hourglass_0);
        lcd->createChar(1, hourglass_1);
```

```
55          lcd->createChar(2, hourglass_2);
56          lcd->createChar(3, hourglass_3);
57          lcd->createChar(4, hourglass_4);
58          lcd->createChar(5, alt);
59          lcd->createChar(6, ant);
60          lcd->createChar(7, sd);
61      #endif
62
63      #if defined(DISPLAY_TYPE_SDD1306_128X64)
64          u8x8_SSD1306 = new U8X8_SSD1306_128X64_NONAME_HW_I2C(U8X8_PIN_NONE, SCL,
    SDA);
65          u8x8_SSD1306->begin();
66          u8x8_SSD1306->setFont(u8x8_font_7x14B_1x2_r);
67      #endif
68
69      #if defined(DISPLAY_TYPE_SDD1306_128X64_lcdgfx)
70          display = new DisplaySSD1306_128x64_I2C(-1);
71          display->begin();
72          //display->setFixedFont(ssd1306xled_font8x16);
73          //display->setFixedFont(ssd1306xled_font6x8);
74          display->setFixedFont(TinyTrackGPS_font8x16);
75      #endif
76      this->clr();
77  }
78
79  void Display::clr(){
80      #if defined(DISPLAY_TYPE_LCD_16X2) || defined(DISPLAY_TYPE_LCD_16X2_I2C)
81          lcd->clear();
82      #endif
83
84      #if defined(DISPLAY_TYPE_SDD1306_128X64)
85          u8x8_SSD1306->clear();
86      #endif
87
88      #if defined(DISPLAY_TYPE_SDD1306_128X64_lcdgfx)
89          display->clear();
90      #endif
91  }
92
93  void Display::print(int vertical, int horizontal, const char text[]){
94      #if defined(DISPLAY_TYPE_LCD_16X2) || defined(DISPLAY_TYPE_LCD_16X2_I2C)
95          lcd->setCursor(vertical, horizontal);
96          this->print(text);
97      #endif
98
99      #if defined(DISPLAY_TYPE_SDD1306_128X64)
100         //u8x8_SSD1306->drawString(vertical, (horizontal*2),text);
101         u8x8_SSD1306->setCursor(vertical, (horizontal*2));
102         this->print(text);
103     #endif
104
105     #if defined(DISPLAY_TYPE_SDD1306_128X64_lcdgfx)
106         display->setTextCursor((vertical*8),(horizontal*16));
107         this->print(text);
108         //display->printFixed((vertical*8),(horizontal*16),text);
109     #endif
110 }
111
112 void Display::print(int line, const char text[]){
113     byte pos = _width -(strlen(text));
```

```
    pos = (pos >> 1);
    this->print((int)pos, line, text);
}

void Display::print(const char text[]){
    #if defined(DISPLAY_TYPE_LCD_16X2) || defined(DISPLAY_TYPE_LCD_16X2_I2C)
        lcd->print(text);
    #endif

    #if defined(DISPLAY_TYPE_SDD1306_128X64)
        u8x8_SSD1306->print(text);
        u8x8_SSD1306->flush();
    #endif

    #if defined(DISPLAY_TYPE_SDD1306_128X64_lcdgfx)
        display->write(text);
    #endif
}

void Display::print(const char text1[], const char text2[]){
    this->print((_screen > 0)?0:1, text1);
    this->print((_screen > 0)?1:2, text2);
}

void Display::print(const char text1[], const char text2[], const char text3[]){
    #if defined(DISPLAY_TYPE_LCD_16X2) || defined(DISPLAY_TYPE_LCD_16X2_I2C)
        this->print(text1, text2);
        delay(750);
        this->clr();
        this->print(0,text3);
    #endif

    #if defined(DISPLAY_TYPE_SDD1306_128X64) ||
defined(DISPLAY_TYPE_SDD1306_128X64_lcdgfx)
        this->print(0, text1);
        this->print(1, text2);
        this->print(2, text3);
    #endif
}

void Display::print(const char text1[], const char text2[], const char text3[],
const char text4[]){
    #if defined(DISPLAY_TYPE_LCD_16X2) || defined(DISPLAY_TYPE_LCD_16X2_I2C)
        this->print(text1,text2);
        delay(750);
        this->print(text3,text4);
    #endif

    #if defined(DISPLAY_TYPE_SDD1306_128X64) ||
defined(DISPLAY_TYPE_SDD1306_128X64_lcdgfx)
        this->print(0, text1);
        this->print(1, text2);
        this->print(2, text3);
        this->print(3, text4);
    #endif
}

void Display::wait_anin(unsigned int t){
    #if defined(DISPLAY_TYPE_LCD_16X2) || defined(DISPLAY_TYPE_LCD_16X2_I2C)
        lcd->setCursor(15,1);
```

```
171         lcd->write((byte)t%5);
172     #endif
173 
174     #if defined(DISPLAY_TYPE_SDD1306_128X64)
175         const char p[4] = {(char)47,(char)45,(char)92,(char)124};
176         u8x8_SSD1306->setCursor((_width-1),6);
177         u8x8_SSD1306->print(p[t%4]);
178         /*
179         static uint8_t hourglass_UP[5][8] = {
0x01,0x1f,0x7f,0xff,0xff,0x7f,0x1f,0x01,
180                                         0x01,0x1f,0x7d,0xf9,0xf9,0x7d,0x1f,0x01,
181                                         0x01,0x1f,0x79,0xf1,0xf1,0x79,0x1f,0x01,
182                                         0x01,0x1f,0x71,0xe1,0xe1,0x71,0x1f,0x01,
183                                         0x01,0x1f,0x61,0x81,0x81,0x61,0x1f,0x01
184                                         };
185 
186         static uint8_t hourglass_DOWN[5][8] =
{0x80,0xf8,0x86,0x81,0x81,0x86,0xf8,0x80,
187                                         0x80,0xf8,0xc6,0xe1,0xe1,0xc6,0xf8,0x80,
188                                         0x80,0xf8,0xe6,0xf1,0xf1,0xe6,0xf8,0x80,
189                                         0x80,0xf8,0xfe,0xf9,0xf9,0xfe,0xf8,0x80,
190                                         0x80,0xf8,0xfe,0xff,0xff,0xfe,0xf8,0x80
191                                         };
192         u8x8_SSD1306->drawTile((_width-1), 6, 1, hourglass_UP[t%5]);
193         u8x8_SSD1306->drawTile((_width-1), 7, 1, hourglass_DOWN[t%5]);
194         */
195     #endif
196 
197     #if defined(DISPLAY_TYPE_SDD1306_128X64_lcdgfx)
198         display->setTextCursor(0,48);
199         display->printChar((char)(t%3)+58);
200     #endif
201 }
202 
203 void Display::print_PChar(byte c) {
204     #if defined(DISPLAY_TYPE_LCD_16X2) || defined(DISPLAY_TYPE_LCD_16X2_I2C)
205         lcd->write(c);
206     #endif
207 
208     #if defined(DISPLAY_TYPE_SDD1306_128X64)
209         static uint8_t PChar_UP[3][8] = { 0x30,0x38,0x3c,0xff,0xff,0x3c,0x38,0x30,
210                                         0x3c,0x02,0x01,0xd9,0xd9,0x01,0x02,0x3c,
211                                         0x78,0x7c,0x6e,0x66,0x66,0x6e,0x7c,0x78
212                                         };
213         static uint8_t PChar_DOWN[3][8] = { 0x00,0x00,0x00,0xff,0xff,0x00,0x00,0x00,
214                                         0x00,0xc0,0xe0,0xff,0xff,0xe0,0xc0,0x00,
215                                         0x7c,0xfc,0xc0,0xf8,0x7c,0x0c,0xfc,0xf8
216                                         };
217         if (c == 5) {
218             u8x8_SSD1306->drawTile(9, 2, 1, PChar_UP[0]);
219             u8x8_SSD1306->drawTile(9, 3, 1, PChar_DOWN[0]);
220         }
221         else if (c == 6) {
222             u8x8_SSD1306->drawTile(11, 0, 1, PChar_UP[1]);
223             u8x8_SSD1306->drawTile(11, 1, 1, PChar_DOWN[1]);
224         }
225         else if (c == 7) {
226             u8x8_SSD1306->drawTile(15, 0, 1, PChar_UP[2]);
227             u8x8_SSD1306->drawTile(15, 1, 1, PChar_DOWN[2]);
228         }
```

```
    #endif

    #if defined(DISPLAY_TYPE_SDD1306_128X64_lcdgfx)
        display->print((char)(c+86));
    #endif
}

void Display::DrawLogo() {
    #if defined(DISPLAY_TYPE_SDD1306_128X64_lcdgfx)
        //display->drawBitmap1(48,24,32,32,Logo_32x32);
        //display->drawBitmap1(32,18,96,16,TinyTrackGPS_96x16);
        this->print(4,0,VERSION);
        this->print(6,1,"^_`a");
        this->print(6,2,"bcde");
    #endif
}

void Display::drawbattery(uint8_t level){
    #if defined(DISPLAY_TYPE_SDD1306_128X64_lcdgfx)
        uint8_t y = 60 - level;
        NanoRect batt = { {122, y} , {125, 60} };
        this->print(14, 2, ",=");
        this->print(14, 3, "+>");
        display->fillRect(batt);
    #endif
}

#if defined(DISPLAY_TYPE_SDD1306_128X64_lcdgfx)
const PROGMEM uint8_t TinyTrackGPS_font8x16[] = {
    0x00, // 0x00 means fixed font type - the only supported by the library
    0x08, // 0x08 = 8 - font width in pixels
    0x10, // 0x10 = 16 - font height in pixels
    0x2b, // Start char. (43)
    // Chars for 'Charge%' text on vertical.
    0x00, 0x00, 0x62, 0x14, 0x74, 0x74, 0x00, 0x00, 0x34, 0x44, 0x44, 0x46, 0x45,
0x35, 0x00, 0x00, // Code for char +
    0x00, 0x64, 0x68, 0x10, 0x2c, 0x4c, 0x00, 0x00, 0x00, 0x00, 0x46, 0xee, 0xa8,
0x6e, 0x20, 0xc0, // Code for char ,
    // Chars numbers and signs.
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x01, 0x01, 0x01,
0x01, 0x01, 0x01, // - 45
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x30, 0x30, 0x00, 0x00,
0x00, 0x00, 0x00, // . 46
    0x40, 0x40, 0x40, 0x40, 0x40, 0x40, 0x40, 0x00, 0x04, 0x04, 0x04, 0x04, 0x04,
0x04, 0x04, 0x00, // '/'->'=' 47
    0x00, 0xE0, 0x10, 0x08, 0x08, 0x10, 0xE0, 0x00, 0x00, 0x0F, 0x10, 0x20, 0x20,
0x10, 0x0F, 0x00, // 0 48
    0x00, 0x10, 0x10, 0xF8, 0x00, 0x00, 0x00, 0x00, 0x00, 0x20, 0x20, 0x3F, 0x20,
0x20, 0x00, 0x00, // 1 49
    0x00, 0x70, 0x08, 0x08, 0x08, 0x88, 0x70, 0x00, 0x00, 0x30, 0x28, 0x24, 0x22,
0x21, 0x30, 0x00, // 2 50
    0x00, 0x30, 0x08, 0x88, 0x88, 0x48, 0x30, 0x00, 0x00, 0x18, 0x20, 0x20, 0x20,
0x11, 0x0E, 0x00, // 3 51
    0x00, 0x00, 0xC0, 0x20, 0x10, 0xF8, 0x00, 0x00, 0x00, 0x07, 0x04, 0x24, 0x24,
0x3F, 0x24, 0x00, // 4 52
    0x00, 0xF8, 0x08, 0x88, 0x88, 0x08, 0x08, 0x00, 0x00, 0x19, 0x21, 0x20, 0x20,
0x11, 0x0E, 0x00, // 5 53
    0x00, 0xE0, 0x10, 0x88, 0x88, 0x18, 0x00, 0x00, 0x00, 0x0F, 0x11, 0x20, 0x20,
0x11, 0x0E, 0x00, // 6 54
```

```
276     0x00, 0x38, 0x08, 0x08, 0xC8, 0x38, 0x08, 0x00, 0x00, 0x00, 0x00, 0x3F, 0x00,
    0x00, 0x00, 0x00, // 7 55
277     0x00, 0x70, 0x88, 0x08, 0x08, 0x88, 0x70, 0x00, 0x00, 0x1C, 0x22, 0x21, 0x21,
    0x22, 0x1C, 0x00, // 8 56
278     0x00, 0xE0, 0x10, 0x08, 0x08, 0x10, 0xE0, 0x00, 0x00, 0x00, 0x31, 0x22, 0x22,
    0x11, 0x0F, 0x00, // 9 57
279     // Chars for wait animation.
280     0x01, 0x1f, 0x7f, 0xff, 0xff, 0x7f, 0x1f, 0x01, 0x80, 0xf8, 0x86, 0x81, 0x81,
    0x86, 0xf8, 0x80, // ':'->wait1 58
281     0x01, 0x1f, 0x79, 0xf1, 0xf1, 0x79, 0x1f, 0x01, 0x80, 0xf8, 0xe6, 0xf1, 0xf1,
    0xe6, 0xf8, 0x80, // ';'->wait2 59
282     0x01, 0x1f, 0x61, 0x81, 0x81, 0x61, 0x1f, 0x01, 0x80, 0xf8, 0xfe, 0xff, 0xff,
    0xfe, 0xf8, 0x80, // '<'->wait3 60
283     // Chars for battery icon.
284     0xfc, 0x02, 0x03, 0x03, 0x03, 0x03, 0x02, 0xfc, 0xff, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0xff, // Code for char =
285     0xff, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xff, 0x7f, 0x40, 0x40, 0x40, 0x40,
    0x40, 0x40, 0x40, 0x7f, // Code for char >
286     // Chars for display space (' ') and 'm' char.
287     0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, // ? ->' ' 63
288     0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x00, 0x20, 0x3F, 0x20, 0x00, 0x3F,
    0x20, 0x00, 0x3F, // @ ->'m' 64
289     // Letters chars for UTM Zone.
290     0x00, 0x00, 0xC0, 0x38, 0xE0, 0x00, 0x00, 0x00, 0x20, 0x3C, 0x23, 0x02, 0x02,
    0x27, 0x38, 0x20, // A 33
291     0x08, 0xF8, 0x88, 0x88, 0x88, 0x70, 0x00, 0x00, 0x20, 0x3F, 0x20, 0x20, 0x20,
    0x11, 0x0E, 0x00, // B 34
292     0xC0, 0x30, 0x08, 0x08, 0x08, 0x08, 0x38, 0x00, 0x07, 0x18, 0x20, 0x20, 0x20,
    0x10, 0x08, 0x00, // C 35
293     0x08, 0xF8, 0x08, 0x08, 0x08, 0x10, 0xE0, 0x00, 0x20, 0x3F, 0x20, 0x20, 0x20,
    0x10, 0x0F, 0x00, // D 36
294     0x08, 0xF8, 0x88, 0x88, 0xE8, 0x08, 0x10, 0x00, 0x20, 0x3F, 0x20, 0x20, 0x23,
    0x20, 0x18, 0x00, // E 37
295     0x08, 0xF8, 0x88, 0x88, 0xE8, 0x08, 0x10, 0x00, 0x20, 0x3F, 0x20, 0x00, 0x03,
    0x00, 0x00, 0x00, // F 38
296     0xC0, 0x30, 0x08, 0x08, 0x08, 0x38, 0x00, 0x00, 0x07, 0x18, 0x20, 0x20, 0x22,
    0x1E, 0x02, 0x00, // G 39
297     0x08, 0xF8, 0x08, 0x00, 0x00, 0x08, 0xF8, 0x08, 0x20, 0x3F, 0x21, 0x01, 0x01,
    0x21, 0x3F, 0x20, // H 40
298     0x00, 0x08, 0x08, 0xF8, 0x08, 0x08, 0x00, 0x00, 0x00, 0x20, 0x20, 0x3F, 0x20,
    0x20, 0x00, 0x00, // I 41
299     0x00, 0x00, 0x08, 0x08, 0xF8, 0x08, 0x08, 0x00, 0xC0, 0x80, 0x80, 0x80, 0x7F,
    0x00, 0x00, 0x00, // J 42
300     0x08, 0xF8, 0x88, 0xC0, 0x28, 0x18, 0x08, 0x00, 0x20, 0x3F, 0x20, 0x01, 0x26,
    0x38, 0x20, 0x00, // K 43
301     0x08, 0xF8, 0x08, 0x00, 0x00, 0x00, 0x00, 0x00, 0x20, 0x3F, 0x20, 0x20, 0x20,
    0x20, 0x30, 0x00, // L 44
302     0x08, 0xF8, 0xF8, 0x00, 0xF8, 0xF8, 0x08, 0x00, 0x20, 0x3F, 0x00, 0x3F, 0x00,
    0x3F, 0x20, 0x00, // M 45
303     0x08, 0xF8, 0x30, 0xC0, 0x00, 0x08, 0xF8, 0x08, 0x20, 0x3F, 0x20, 0x00, 0x07,
    0x18, 0x3F, 0x00, // N 46
304     0xE0, 0x10, 0x08, 0x08, 0x08, 0x10, 0xE0, 0x00, 0x0F, 0x10, 0x20, 0x20, 0x20,
    0x10, 0x0F, 0x00, // O 47
305     0x08, 0xF8, 0x08, 0x08, 0x08, 0x08, 0xF0, 0x00, 0x20, 0x3F, 0x21, 0x01, 0x01,
    0x01, 0x00, 0x00, // P 48
306     0xE0, 0x10, 0x08, 0x08, 0x08, 0x10, 0xE0, 0x00, 0x0F, 0x18, 0x24, 0x24, 0x38,
    0x50, 0x4F, 0x00, // Q 49
307     0x08, 0xF8, 0x88, 0x88, 0x88, 0x88, 0x70, 0x00, 0x20, 0x3F, 0x20, 0x00, 0x03,
    0x0C, 0x30, 0x20, // R 50
```

```
    0x00, 0x70, 0x88, 0x08, 0x08, 0x08, 0x38, 0x00, 0x00, 0x38, 0x20, 0x21, 0x21, 0x22, 0x1C, 0x00, // S 51
    0x18, 0x08, 0x08, 0xF8, 0x08, 0x08, 0x18, 0x00, 0x00, 0x00, 0x20, 0x3F, 0x20, 0x00, 0x00, 0x00, // T 52
    0x08, 0xF8, 0x08, 0x00, 0x00, 0x08, 0xF8, 0x08, 0x00, 0x1F, 0x20, 0x20, 0x20, 0x20, 0x1F, 0x00, // U 53
    0x08, 0x78, 0x88, 0x00, 0x00, 0xC8, 0x38, 0x08, 0x00, 0x00, 0x07, 0x38, 0x0E, 0x01, 0x00, 0x00, // V 54
    0xF8, 0x08, 0x00, 0xF8, 0x00, 0x08, 0xF8, 0x00, 0x03, 0x3C, 0x07, 0x00, 0x07, 0x3C, 0x03, 0x00, // W 55
    0x08, 0x18, 0x68, 0x80, 0x80, 0x68, 0x18, 0x08, 0x20, 0x30, 0x2C, 0x03, 0x03, 0x2C, 0x30, 0x20, // X 56
    0x08, 0x38, 0xC8, 0x00, 0xC8, 0x38, 0x08, 0x00, 0x00, 0x00, 0x20, 0x3F, 0x20, 0x00, 0x00, 0x00, // Y 57
    0x10, 0x08, 0x08, 0x08, 0xC8, 0x38, 0x08, 0x00, 0x20, 0x38, 0x26, 0x21, 0x20, 0x20, 0x18, 0x00, // Z 58
    // Personalized simbols for display info.
    0x30, 0x38, 0x3c, 0xff, 0xff, 0x3c, 0x38, 0x30, 0x00, 0x00, 0x00, 0xff, 0xff, 0x00, 0x00, 0x00, // '['->altitud 91
    0x3c, 0x02, 0x01, 0xd9, 0xd9, 0x01, 0x02, 0x3c, 0x00, 0xc0, 0xe0, 0xff, 0xff, 0xe0, 0xc0, 0x00, // '\'->antena 92
    0x78, 0x7c, 0x6e, 0x66, 0x66, 0x6e, 0x7c, 0x78, 0x7c, 0xfc, 0xc0, 0xf8, 0x7c, 0x0c, 0xfc, 0xf8, // ']'->sd 93
    // Chars as logo. Up 'abcd' down 'efgh'.
    0x00, 0x00, 0x80, 0xC0, 0x60, 0x10, 0x98, 0x4C, 0xF0, 0x1E, 0x03, 0xF0, 0x0C, 0x03, 0x81, 0x00, // char 94
    0x64, 0x26, 0x12, 0x12, 0x0B, 0x09, 0x09, 0x49, 0x00, 0x00, 0x00, 0x00, 0x80, 0xC0, 0x60, 0xB0, // char 95
    0x49, 0x09, 0x09, 0x0B, 0x12, 0x12, 0x26, 0x64, 0x10, 0x18, 0x08, 0xC4, 0x64, 0x1E, 0x07, 0x03, // char 96
    0x48, 0x98, 0x10, 0x60, 0xC0, 0x00, 0x00, 0x00, 0x00, 0x81, 0x03, 0x0C, 0xF0, 0x03, 0x1E, 0xF0, // char 97
    0x0F, 0x78, 0xC0, 0x0F, 0x30, 0xC0, 0x81, 0x00, 0x00, 0x00, 0x01, 0x03, 0x06, 0x08, 0x19, 0x32, // char 98
    0xC0, 0xE0, 0x78, 0x26, 0x23, 0x10, 0x18, 0x08, 0x26, 0x64, 0x48, 0x48, 0xD0, 0x90, 0x90, 0x92, // char 99
    0x0D, 0x06, 0x03, 0x01, 0x00, 0x00, 0x00, 0x00, 0x92, 0x90, 0x90, 0xD0, 0x48, 0x48, 0x64, 0x26, // char 100
    0x00, 0x81, 0xC0, 0x30, 0x0F, 0xC0, 0x78, 0x0F, 0x32, 0x19, 0x08, 0x06, 0x03, 0x01, 0x00, 0x00, // char 101
};
#endif
```